Weak Concurrent Kleene Algebra with Application to Algebraic Verification

Annabelle McIver[1] **Tahiry Rabehaja**[1] **Georg Struth**[3]

[1] Department of Computing
Macquarie University,
Sydney, Australia
Email: {annabelle.mciver,tahiry.rabehaja}@mq.edu.au

[2] Department of Computer Science
University of Sheffield,
United Kingdom,
Email: g.struth@dcs.shef.ac.uk

Abstract

We propose a generalisation of concurrent Kleene algebra [5] that can take account of probabilistic effects in the presence of concurrency. The algebra is proved sound with respect to a model of automata modulo a variant of rooted η-simulation equivalence. Applicability is demonstrated by algebraic treatments of two examples: algebraic may testing and Rabin's solution to the choice coordination problem.

1 Introduction

Kleene algebra generalises the language of regular expressions and, as a basis for reasoning about programs and computing systems, it has been used in applications ranging from compiler optimisation, program refinement, combinatorial optimisation and algorithm design [2, 6, 7, 8, 10]. A number of variants of the original axiom system and language of Kleene algebra have extended its range of applicability to include probability [12] with the most recent being the introduction of a concurrency operator [5]. Main benefits of the algebraic approach are that it captures some essential aspects of computing systems in a simple and concise way and that the calculational style of reasoning it supports is very suitable for automated theorem proving.

In this paper we continue this line of work and propose *weak concurrent Kleene algebra*, which extends the abstract probabilistic Kleene algebra [12] with the concurrency operator of concurrent Kleene algebra [5] and thus supports reasoning about concurrency in a context of probabilistic effects. This extension calls for a careful evaluation of the axiom system so that it accurately accounts for the interactions of probabilistic choice, nondeterministic choice and the treatment of concurrency. For example probabilistic Kleene algebra accounts for the presence of probability in the *failure* of the original distributive law $x(y+z) = xy + xz$ which is also absent in most process algebras. That is because when the terms x, y, z are interpreted as probabilistic programs, with xy meaning "first execute x and then y" and $+$ interpreted as a nondeterministic choice, the expression on the left hand side exhibits a greater range of nondeterminism than the right in the case that x includes probabilistic behaviours. For example if x is interpreted as a program which flips a bit with probability $1/2$ then the following nondeterministic choice in $y + z$ can always be resolved so that y is executed *if and only if* the bit was indeed flipped. This is not a behaviour amongst those described by $xy + xz$, where the nondeterminism is resolved before the bit is flipped and therefore its resolution is unavoidably independent of the flipping. Instead, in contexts such as these, distributivity be replaced by a weaker law:

$$\text{Sub-distributivity:} \qquad xy + xz \;\leq\; x(y+z)\,. \quad (1)$$

Elsewhere [9] we show that this weakening of the original axioms of Kleene algebra results in a complete system relative to a model of nondeterministic automata modulo simulation equivalence.

The behaviour of the concurrency operator of concurrent Kleene algebra [5] is captured in particular by the *Interchange law*:

$$(x \| y)(u \| v) \;\leq\; (xu) \| (yv)$$

which expresses that there is a lesser range of nondeterministic executions on the left where, for example, the execution of u is constrained to follow a complete execution of x run concurrently with y but on the right it is not.

Our **first contribution** is the construction of a concrete model of abstract probabilistic automata (where the probability is at the action level) over which to interpret terms composed of traditional Kleene algebra together with concurrent composition. In this interpretation, each term represents an automaton. For example in Equation (1), x, y and z are automata and so is $xy + xz$. We show that the axiom system of concurrent Kleene algebra weakened to allow for the presence of probability is sound with respect to those probabilistic automata. Our use of probabilistic automata is similar to models where the resolution of probability and nondeterminism can be interleaved; concurrent composition of automata models CSP synchronisation [4] in that context. Finally we use a notion of rooted η-simulation to interpret the inequality $\leq$ used in algebraic inequations.

Our **second contribution** is to explore some applications of our axiomatisation of weak concurrent Kleene algebra, to explain our definition of rooted η-simulation in terms of may testing [14], and to demonstrate the proof system on Rabin's distributed consensus protocol [15].

This work has been supported by the iMQRES grant from Macquarie University.

One of the outcomes of this study is to expose the tensions between the various aspects of system execution. Some of the original concurrent Kleene algebra axioms [5] required for the concurrency operator now fail to be satisfiable in the presence of probabilistic effects and synchronisation supported by the interchange law. For example, the term 1 from Kleene algebra (interpreted as "do nothing") can no longer be a neutral element for the concurrency operator $\|$ — we only have the specific equality $1\|1 = 1$ and not the more general $1\|x = x$. In fact we chose to preserve the full interchange law in our choice of axioms because it captures so many notions of concurrency already including exact parallel and synchronisation, suggesting that it is a property about general concurrent interactions.

A feature of our approach is to concentrate on broad algebraic structures in order to understand how various behaviours interact rather than to study precise quantitative behaviours. Thus we do not include an explicit probabilistic choice operator in the signature of the algebra — probability occurs explicitly only in the concrete model as a special kind of asynchronous probabilistic action combined with internal events (events that the environment cannot access). This allows the specification of complex concurrent behaviour to be simplified using applications of weak distributivity embodied by Equation (1) and/or the interchange law as illustrated by our case study. Finally we note that the axiomatisation we give is entirely in terms of first-order expressions and therefore is supported by first-order reasoning. Thus all of our algebraic proofs has been implemented within the Isabelle/HOL theorem proving environment. These proof can be found in a repository of formalised algebraic theorems. [1]

In Section 2 we explore the axiomatisation of the new algebra. It is essentially a mixture of probabilistic and concurrent Kleene algebras. Sections 3 and 4 are devoted to showing the consistency of our approach. A concrete model based on automata and η-simulation is constructed. In section 5, we compare our approach with probabilistic automata (automata that exhibit explicit probability) and probabilistic simulation. We conclude that, up to some constraint, the concrete model is a very special case of that more general model. In sections 6 and 7, we present some applications, in particular an algebraic version of may testing is studied and variations of the specification of Rabin's protocol are explored.

In this paper x, y, etc represent algebraic expressions or variables. Terms are denoted s, t, etc. Letters a, b, etc stand for actions and τ represents an internal action. An automaton associated with a term or an expression is usually denoted by the same letter. Other notation is introduced as we need it.

In this extended abstract we can only explain the main properties of weak concurrent Kleene algebra and sketch the construction of the automaton model. Detailed constructions and proofs of all statements in this paper can be found in an appendix.

2 Axiomatisation

A Kleene algebra is a structure that encodes algebraically the sequential behaviour of a system. It is generally presented in the form of an idempotent [2] semiring structure $(K, +, \cdot, 0, 1)$ where $x \cdot y$ (sequential composition) is sometimes written using juxtaposition xy in expressions. The term 0 is the neutral element of $+$ and 1 is the neutral element of $\cdot$. The semiring is then endowed with a unary Kleene star $*$ representing finite iteration to form a Kleene algebra. This operator is restricted by the following axioms:

$$\text{Left unfold:} \quad 1 + xx^* = x^*, \tag{2}$$

$$\text{Left induction:} \quad xy \leq y \Rightarrow x^*y \leq y, \tag{3}$$

where $x \leq y$ if and only if $x+y = y$. In the sequel our interpretations will be over a version of probabilistic automata. In particular we will interpret $\leq$ and $=$ as η-simulations.

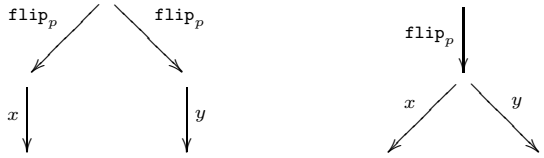
Often, the dual of (2-3) i.e. $1 + x^*x = x^*$ and $yx \leq y \Rightarrow yx^* \leq y$ are also required. However, (2) and (3) are sufficient here and the dual laws follow from continuity of sequential composition for finite automata.

In a Kleene algebra, the semiring structure supports two distributivity laws:

$$\text{Left distributivity:} \quad xy + xz = x(y + z), \tag{4}$$

$$\text{Right distributivity:} \quad (x + y)z = xz + yz. \tag{5}$$

Equation (4) however is not valid in the presence of probability. For example, compare the behaviour of probabilistic choice in the diagrams below. Here, $\mathtt{flip}_p$ denotes the process that flips a p-biased coin, which we can represent by a probabilistic automaton (details are given in Section 3). In the right diagram,



the choice between a and b can be based on the outcome of the coin flip but such resolution is not possible in the left-hand diagram. We express the greater range of possible outcomes by the general inequation (1), specifically here it becomes

$$(\mathtt{flip}_p)y + (\mathtt{flip}_p)z \leq (\mathtt{flip}_p)(y + z). \text{ [3]} \tag{6}$$

As mentioned above, the zero of a Kleene algebra satisfies:

$$\text{Left annihilation:} \quad 0x = 0, \tag{7}$$

$$\text{Right annihilation:} \quad x0 = 0. \tag{8}$$

In our interpretation that includes concurrency, we assume that 0 captures *deadlock*. However, axiom (8) is no longer appropriate because we should be able to differentiate between the process doing an action and deadlocking from a process that is just deadlocked.

Definition 1. *A weak probabilistic Kleene algebra is a structure $(K, +, \cdot, *, 0, 1)$ that satisfies the axioms of Kleene algebra except there is no left distributivity (it is replaced by (1)) and Equation (8) does not hold generally.*

A concurrency operator was added to Kleene algebra by Hoare et al [5]. Our concurrency operator $\|$

[1] `http://staffwww.dcs.shef.ac.uk/people/G.Struth/isa/`

[2] Idempotence refers to the operation $+$ i.e. $x + x = x$.

[3] We have abused notation in this example by using $\mathtt{flip}_p$ to represent both an action and an automaton which performs that action.

satisfies the following standard axioms:

$$\begin{aligned} \text{Associativity:} \quad & x\|(y\|z) = (x\|y)\|z, && (9)\\ \text{Commutativity:} \quad & x\|y = y\|x, && (10)\\ \text{One-idempotence:} \quad & 1\|1 = 1. && (11) \end{aligned}$$

In [5], $\|$ satisfies the identity $1\|x = x$ which we do not have here because in the concrete model, we will interpret $\|$ as the synchronisation operator found in CSP [4]. However, we still maintain the instance of that law in the special case $x = 1$ (see axiom (11)) where 1 is interpreted as "do nothing".

Next we have the axioms dealing the interaction of $\|, +$ and $\cdot$.

$$\begin{aligned} \text{Monotonicity :} \quad & x\|y + x\|z \leq x\|(y+z) && (12)\\ \text{Interchange-law:} \quad & (x\|y)(u\|v) \leq (xu)\|(yv) && (13) \end{aligned}$$

The interchange law is the most interesting axiom of concurrent Kleene algebra. In fact it allows the derivation of many properties involving $\|$. To illustrate this in the probabilistic context, consider a probabilistic vending machine VM which we describe as the expression

$$\mathtt{VM} \;=\; \mathtt{coin}\cdot\mathtt{flip}_p\cdot(\tau_h\cdot(\mathtt{tea}+1)+\tau_t\cdot(\mathtt{coffee}+1))$$

where $\mathtt{coin}, \mathtt{tea}, \mathtt{coffee}, \tau_h, \tau_t$ and $\mathtt{flip}_p$ are all represented by automata. That is the vending machine accepts a coin and then decides internally whether it will enable the button coffee or tea. The decision is determined by the action $\mathtt{flip}_p$ [4] which (as explained later) enables either τ_h or τ_t. The actions τ_t and τ_h are internal and the user cannot access them. Now, a user who wants to drink tea is specified as

$$\mathtt{U} \;=\; \mathtt{coin}\cdot(\mathtt{tea}+1).$$

The system becomes $\mathtt{U}\|\mathtt{VM}$ where the concurrent operation is CSP like and synchronises on $\mathtt{coin}, \mathtt{tea}$ and $\mathtt{coffee}$. The interchange law together with the other axioms and some system assumptions imply the following inequation:

$$\mathtt{U}\|\mathtt{VM} \;\geq\; \mathtt{coin}\cdot\mathtt{flip}_p\cdot(\tau_h\cdot(\mathtt{tea}+1)+\tau_t) \qquad (14)$$

which is proved automatically in our repository. In other words, the user will only be satisfied with *probability at least p* since the right-hand side equation says that the tea action can only be enabled provided that τ_h is enabled, and in turn that is determined by the result of the $\mathtt{flip}_p$ action.

Now we are ready to define our algebra.

Definition 2. *A weak concurrent Kleene algebra is a weak probabilistic Kleene algebra $(K, +, \cdot, *, 0, 1)$ with a concurrency operator $\|$ satisfying (9-13)*

We assume the operators precedence $* < \cdot < \| < +$.

Proposition 3. *Let s, t be terms, the following equations holds in weak concurrent Kleene algebra.*

1. *All the operators are monotonic.*
2. $(s^*\|t^*)^* = s^*\|t^*$.
3. $(s\|t)^* \leq s^*\|t^*$.
4. $(s+t)^* = (s^*t^*)^*$.

[4] i.e. the automaton that performs a $\mathtt{flip}_p$ action.

3 Concrete Model

3.1 Semantic Space

We use nondeterministic automata to construct a concrete model. An automaton is denoted by a tuple

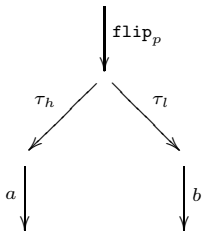
$$(P, \longrightarrow, i, F)$$

where P is a set of states. The set $\longrightarrow \subseteq P \times \Sigma \times P$ is a transition relation and we write $x \xrightarrow{a} y$ when there is a transition, labelled by a, from state x to state y. The alphabet Σ is left implicit and considered to be fixed for every automaton. The state $i \in P$ is the initial state and $F \subseteq P$ is the set of final states of the automaton. In the sequel, we will denote an automaton $(P, \longrightarrow, i, F)$ by its set of states P when no confusion is possible.

The actions in the alphabet Σ are categorised into three kinds:

- *internal*: actions that will be "ignored" by the simulation relation (as in τ_h and τ_t). Internal actions are never synchronised by $\|$.
- *external*: actions that *can* be synchronised. Probabilistic actions are external (as in $\mathtt{flip}_p$) but they are *never* synchronised.
- *synchronised*: external actions that will be synchronised when applying $\|$ (as in $\mathtt{coin}, \mathtt{tea}$ and $\mathtt{coffee}$). These actions are determined by a set of external actions A. More specifically, $\|$ refers to ${}_A\|$ which we assume is fixed and given beforehand.

The special case of probabilistic choice is modelled by combining probabilistic and internal actions. That is a process that does a with probability p and does b with probability $1-p$ is interpreted as the following automaton where $\mathtt{flip}_p \in \Sigma$ represents the action



of flipping a p-biased coin which produces head with probability p and tail with probability $1-p$. The internal actions τ_t and τ_h are enabled according to the result of $\mathtt{flip}_p$. Hence only one of τ_h and τ_t will be enabled just after the coin flip. Since τ_t and τ_h are internal actions, the choice is internal and based upon the outcome of $\mathtt{flip}_p$. The important facts here are that the choice after $\mathtt{flip}_p$ is internal so could be based on the probabilistic outcome of $\mathtt{flip}_p$ and that the environment cannot interfere with that choice. These two behavioural characteristics are what we consider to be the most general features of probability in a concurrent setting and they are those which we axiomatise and record in our concrete model.

Next, we impose some conditions on the automata to ensure soundness.

- reachability: every state of the automaton is reachable by following a finite path from the initial state.

- initiality: there is no transition that leads to the initial state. This means that a^* corresponds to the automata associated to $1 + aa^*$ rather than a self loop labeled by $a \in \Sigma$.

We denote by **Aut** the set of automata satisfying these two conditions. The next step is to define the operators that act on **Aut**. We use the standard inductive construction found in [1, 17, 9] and the diagrams illustrating the constructions are given in the appendix.

Deadlock: 0
This is the automaton that has only one state, namely the initial state, and no transition at all. It is the tuple $(\{i\}, \emptyset, i, \emptyset)$.

Skip: 1
This is the automaton that has only one state i which is both initial and final. This automaton has no transition i.e. is denoted by $(\{i\}, \emptyset, i, \{i\})$.

Single action:
The automata associated with a is $i \xrightarrow{a} \circ$ where i is the initial state and $\circ$ is a final state. It is the tuple $(\{i, \circ\}, \{i \xrightarrow{a} \circ\}, i, \{\circ\})$.

Addition: $P + Q$
This automaton is obtained using the standard construction of identifying the initial states of P and Q. (This is possible due to the initiality property.)

Multiplication: PQ **(or** $P \cdot Q$**)**
This automaton is constructed in the standard way of identifying copies of the initial state of Q with final states of P.

Concurrency: $P_A\|Q$
This automaton is constructed as in CSP [4]. It is a sub-automaton of the Cartesian product of P and Q. The initial state is (i_P, i_Q) and final states are reachable elements of $F_P \times F_Q$. Notice that the set A never contains probabilistic actions. Further explanation about $_A\|$ is given below.

Kleene star: P^*
This automaton is the result of repeating P allowing a successful termination after each (possibly empty) full execution of P. The initial state of P^* is final and copies of the initial state of P are identified with the final states of P.

All automata begin with an initial state and end in some final or deadlock state. Our main use of final states is in the construction of sequential composition and Kleene star.

The concurrency operator $_A\|$ synchronises transitions labeled by an action in A and interleaves the others (including internal transitions). As in CSP, a synchronised transition waits for a corresponding synchronisation action from the other argument of $_A\|$. This is another reason we do not have $1_{\{a\}}\|P = P$ because if $P = i_P \xrightarrow{a} \circ$ and i_P is not a final state, then

$$1_{\{a\}}\|P = (\{(i, i_P)\}, \emptyset, (i, i_P), \emptyset) = 0.$$

Proposition 4. *These operations of weak concurrent Kleene algebra are well defined on* **Aut** *that is if* $P, Q \in$ **Aut** *then* $P + Q, PQ, P_A\|Q$ *and* P^* *are elements of* **Aut**.

The proof consists of checking that $P + Q, PQ, P\|Q$ and P^* satisfy the reachability and initiality conditions whenever P and Q satisfy the same conditions. (See Proposition 20 in the appendix).

In the sequel, whenever we use an unframed concurrency operator $\|$, we mean that the frame A has been given and remains fixed.

3.2 Equivalence

The previous subsection has given us the objects and operators needed to construct our concrete model. Next we turn to the interpretation of equality for our concrete interpretation.

Following the works found in [1, 9, 13], we again use a simulation-like relation to define valid equations in the concrete model. More precisely, due to the presence of internal actions, we will use an η-*simulation* as the basis for our equivalence.

Before we give the definition of simulation, we need the following notation. Given the state x and y, we write $x \Rightarrow y$ if there exists a path, possibly empty, from x to y such that it is labelled by internal actions only. This notation is also used in [17] with the same meaning.

Definition 5. *Let* P, Q *be automata, a relation* $S \subseteq P \times Q$ *(or* $S : P \to Q$*) is called* η***-simulation*** *if*

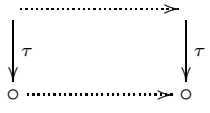
- $(i_P, i_Q) \in S$,
- *if* $(x, y) \in S$ *and* $x \xrightarrow{a} x'$ *then*
  a) *if* a *is internal then there exits* y' *such that* $y \Rightarrow y'$ *and* $(x', y') \in S$,
  b) *if* a *is external then there exists* y_1 *and* y' *in* Q *such that* $y \Rightarrow y_1 \xrightarrow{a} y'$ *and* $(x, y_1) \in S$ *and* $(x', y') \in S$.
- *if* $(x, y) \in S$ *and* $x \in F_P$ *then* $y \in F_Q$.

A simulation S *is* ***rooted*** *if* $(i_P, y) \in S$ *implies* $y = i_Q$. *If there is a rooted simulation from* P *to* Q *then we say that* P *is simulated by* Q *and we write* $P \leq Q$. *Two processes* P *and* Q *are* ***simulation equivalent*** *if* $P \leq Q$ *and* $Q \leq P$, *and we write* $P \equiv Q$. *In the sequel, rooted any* η*-simulation will be referred simply as a simulation.*

Relations satisfying Definition 5 are also η-simulation in the sense of [17] where property (a) is replaced by:

$$\text{if } a \text{ is internal then } (x', y) \in S. \tag{15}$$

The identity relation (drawn as dotted arrow) in the following diagram is a simulation relation satisfying Definition 5, but it is not a simulation in the sense of [17]. We need the identity relation to be a simulation here because in our proof of soundness, more complex simulations are constructed from identity relations.



Proposition 6. *The following statements hold.*

1. *The relational composition of two rooted η-simulations is again a rooted η-simulation. That is, if S, T are rooted η-simulations then $S \circ T$ is also a rooted η-simulation, where $\circ$ denotes relational composition.*

2. *The simulation relation $\leq$ is a preorder on* **Aut**.

Proposition 6 is proven in Proposition 21 of the appendix.

Therefore, $\equiv$ as determined by Definition 5 is an equivalence. In fact, we prove in the following proposition that it is a congruence with respect to $+$.

Proposition 7. *The equivalence relation $\equiv$ is a congruence with respect to $+$ and $P \leq Q$ iff $P + Q \equiv Q$.*

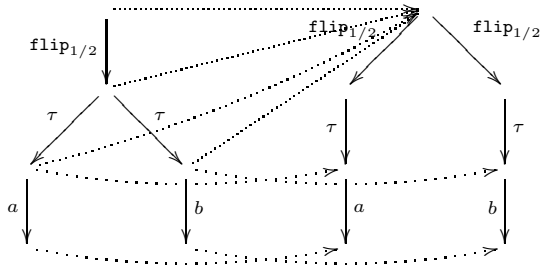
The proof adapts and extends the one found in [17] and the specialised version for our case is Proposition 22 in the appendix.

It is well documented that η-simulation is not a congruence without the rootedness condition [17]. A typical example is given by the expressions $\tau a + \tau b$ and $\tau(a + b)$. The automata associated to these expressions are equivalent under non-rooted η-simulation.

The manipulation of probabilistic actions is also an important facet of our model. We assume that probabilistic actions are not synchronised and in that respect they are similar to internal actions. However probabilistic actions cannot be treated as internal as the following examples illustrates. Consider the action $\mathtt{flip}_{1/2}$ which flips a fair coin. If $\mathtt{flip}$ is an internal action then the inequality

$$(\mathtt{flip}_{1/2})(\tau a + \tau b) \leq (\mathtt{flip}_{1/2})\tau a + (\mathtt{flip}_{1/2})\tau b$$

would be valid when interpreted in the concrete model. In other words, we would have the following simulation:



But this relationship (which implies distributivity of $\mathtt{flip}_p$ through $+$) does not respect the desired behaviour of probability which, as we explained earlier, satisfies only a weaker form of distributivity. Whence, we assume that probabilistic actions such as $\mathtt{flip}_{1/2}$ are among the external actions which will never be synchronised.

4 Soundness

In this section, we prove that the set **Aut** endowed with the operators defined in Subsection 3.1 modulo rooted η-simulation equivalence (Subsection 3.2) forms a weak concurrent Kleene algebra.

The first part is to prove that **Aut** is a weak probabilistic Kleene algebra.

Proposition 8. $(\mathbf{Aut}, +, \cdot, *, 0, 1)$ *is a weak probabilistic Kleene algebra.*

The proof consists of detailed verifications of the axioms for weak probabilistic Kleene algebra (see Proposition 23 in the appendix).

The second part consists of proving that $\|$ satisfies the equations (9-13). Associativity depends heavily on the fact that both concurrent compositions involved in $x\|y\|z$ have the same frame set. For instance, let $\Sigma = \{a, b, c\}$. The identities

$$(a_{\{a\}}\|b)_{\{c\}}\|a = ab0 + ba0$$

and

$$a_{\{a\}}\|(b_{\{c\}}\|a) = ab + ba$$

are valid in the concrete model. Hence, the first process will always go into a deadlock state though the second one will always terminate successfully. Therefore, to have associativity, the concurrency operator must have a fixed frame.

Proposition 9. $(\mathbf{Aut}, +, \cdot_A\|, 1)$ *satisfies equations (9- 13) modulo rooted η-simulation equivalence for any set of synchronisable actions $A \subseteq \Sigma$ (i.e. no probabilistic actions).*

Associativity is mainly a consequence of the fact that there is only one frame for $\|$. The other axioms need to be checked thoroughly (see Proposition 24).

Our soundness result directly follows from these two propositions.

Theorem 10. $(\mathbf{Aut}, +, \cdot_A\|, *, 0, 1)$ *is a weak concurrent Kleene algebra for any set of synchronisable actions $A \subseteq \Sigma$.*

In this theorem, the frame A is fixed beforehand. In other words, a model of weak concurrent Kleene algebra is constructed for each possible choice of A. In particular, if A is empty then the concurrency operator is interleaving all actions i.e. no actions are synchronised. This particular model satisfies the identity $1_\emptyset\|x = x$ of the original concurrent Kleene algebra found in [5].

The sequential and concurrent composition actually have stronger properties in the concrete model. If we consider finite automata only — automata with finitely many states and transitions— then we show that these two operators are *conditionally Scott continuous* in the sense of [9] (see Proposition 25 and 27 in the appendix).

5 Relationship to Probabilistic Processes

Firstly, it is shown in [11] that a probabilistic choice $a_p\oplus b$ simulates the nondeterministic choice $a + b$. A similar result also holds in our setting. In the absence of internal transitions, simulation has been also defined elsewhere [1, 17, 9] which we will refer to as strong simulation. Recall that $(\mathtt{flip}_p)a + (\mathtt{flip}_p)b \leq (\mathtt{flip}_p)(a + b)$ is a general property of probabilistic Kleene algebra [12] so it is valid under strong simulation equivalence [1, 9]. Due to the absence of internal actions, the middle part of the diagram of Figure 1 does not exist with respect to strong simulation equivalence.

In the context of Definition 5, the right-hand simulation of Figure 1 is the refinement of probabilistic choice by nondeterminism. This example gives an explicit distinction between $(\mathtt{flip}_p)(a + b)$ and $(\mathtt{flip}_p)a + (\mathtt{flip}_p)b$ by considering the fact that the choice in $(\mathtt{flip}_p)(a + b)$ can depend on the probabilistic outcome of $(\mathtt{flip}_p)$, but this is not the case for $(\mathtt{flip}_p)a + (\mathtt{flip}_p)b$.

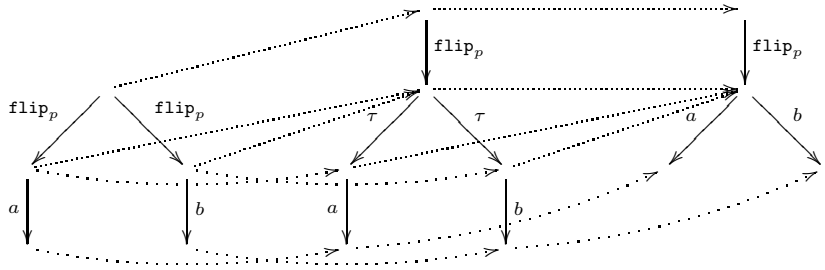


Figure 1: Refinements between probabilistic choice and nondeterminism.

Secondly, we discuss about the relationship between our concrete model and probabilisitic automata. Remind that our interpretation of probability lies in the use of actions that implicitly contain probabilistic information. In its most general form, a probabilistic choice between n possibilities can be written as

$$\texttt{flip}_{p_1,\ldots,p_n} \cdot (\tau_1 \cdot a_1 + \ldots + \tau_n \cdot a_n)$$

where $\sum_i p_i = 1$. In this algebraic expression, we implicitly ensure that each guard τ_i is enabled with a corresponding probability p_i. Therefore, if these τ_i's are not found directly after the execution of the probabilistic action then matching them with the corresponding p_i becomes a difficult task. We call p-automaton [5] a transition system as per the definition of Subsection 3.1 such that if a probabilistic action has associated τ transitions then all of them follow that action directly.

Another complication also arises from the use of these τ_i's. Consider the following two processes

$$\texttt{flip}_{p_1,p_2} \cdot (\tau_1 \cdot a + \tau_2 \cdot b)$$

and

$$\texttt{flip}_{p_1,p_2} \cdot (\tau_1 \cdot b + \tau_2 \cdot a)$$

where $p_1 + p_2 = 1$. We can construct a (bi)simulation relation between the corresponding automata though the probabilities of doing an a are different. Hence we need to modify the definition of η-simulation (Definition 5) to account for these particular structure.

Definition 11. *A p-simulation S between two p-automata P, Q is a η-simulation such that if*

- $x \xrightarrow{\texttt{flip}_{p_1,\ldots,p_n}} x' \xrightarrow{\tau_i} x_i''$ *is a transition in P,*
- $y \xrightarrow{\texttt{flip}_{p_1,\ldots,p_n}} y' \xrightarrow{\tau_i} y_i''$ *is a transition in Q,*
- *and $(x, y) \in S$*

then $(x_i'', y_i'') \in S$, for each $i = 1, \ldots, n$.

This definition ensures that the probability of doing a certain action from y is greater than doing that action from x. With similar proofs as in the previous Sections, we can show that the set of p-automata modulo p-simulation forms again a weak concurrent Kleene algebra. We denote p-**Aut** the set of p-automata modulo p-simulation.

We will now show that this definition is a very special case of probabilistic simulation on probabilistic automata. To simplify the comparison, we assume that τ transitions occur only as part of these probabilistic choices in p-automata.

[5] The name p-automata describes probabilistic automata and as we will see later on, there is a relationship between the two of them.

Definition 12. *A probabilistic automaton is defined as a tuple $(P, \longrightarrow, \Delta, F)$ where P is a set of states, $\longrightarrow$ is a set of labelled transitions from state to distributions [6] of states i.e. $\longrightarrow \subseteq P \times \Sigma \times \mathcal{D}P$, Δ is the initial distribution and $F \subseteq P$ is a set of final states.*

The notion of simulation also exists for probabilistic automata [16] and, in particular, simulation and failure simulation is discussed in [3] where they are proven to be equivalent to may and must testing respectively.

To give a proper definition of probabilistic simulation, we need the following notations which are borrowed from [3] and [17]. Given a relation $R \subseteq P \times \mathcal{D}Q$, the lifting of R is a relation $\hat{R} \subseteq \mathcal{D}P \times \mathcal{D}Q$ such that $\phi \hat{R} \psi$ iff:

- $\phi = \sum_x p_x \delta_x$, [7]
- for each $x \in \text{supp}(\phi)$ (the support of ϕ) there exists $\psi_x \in \mathcal{D}Q$ such that $x R \psi_x$,
- $\psi = \sum_x p_x \psi_x$.

Similarly, the lifting of a transition relation $\xrightarrow{\tau}$ is denote $\xrightarrow{\hat{\tau}}$ whose reflexive transitive closure is denote $\stackrel{\hat{\tau}}{\Longrightarrow}$. For each external action a, we write $\stackrel{\hat{a}}{\Longrightarrow}$ for the sequence $\stackrel{\hat{\tau}}{\Longrightarrow}\xrightarrow{a}$.

Definition 13. *A probabilistic simulation S between two probabilistic automata P and Q is a relation $S \subseteq R \times \mathcal{D}Q$ such that:*

- $(\Delta_P, \Delta_Q) \in \hat{S}$,
- *if $(x, \psi) \in S$ and $x \xrightarrow{a} \phi$ then there exists $\psi' \in \mathcal{D}Q$ such that $\psi \stackrel{\hat{a}}{\Longrightarrow} \psi'$ and $(\phi, \psi') \in \hat{S}$ (for every $a \in \Sigma \cup \{\tau\}$).*
- *if $x \in F_P$ and $(x, \psi) \in S$ then $\text{supp}(\psi) \subseteq F_Q$.*

we denote by **ProbAut** the set of probabilistic automata modulo simulation equivalence.

We can now construct a mapping $\epsilon : p\text{-}\mathbf{Aut} \to \mathbf{ProbAut}$ such that each instance of structure similar to $\texttt{flip}_{p_1,\ldots,p_n} \cdot (\tau_1 \cdot a_1 + \ldots + \tau_n \cdot a_n)$ is collapsed into probabilistic transitions. More precisely, let $P \in p\text{-}\mathbf{Aut}$ and $\longrightarrow$ be its transition relation. The automaton $\epsilon(P)$ has the same state space as P (up to accessibility with respect to the transitions of $\epsilon(P)$). The initial distribution of $\epsilon(P)$ is δ_{i_P} and the set of final states of $\epsilon(P)$ is F_P again [8] .

[6] We assume that all distributions are finitely supported.

[7] We denote by δ_x the point distribution concentrated on x.

[8] Notice that by assuming the structure $\texttt{flip}_{p_1,\ldots,p_n} \cdot (\tau_1 \cdot a_1 + \ldots + \tau_n \cdot a_n$, the state between the flip action the corresponding τ transitions is never a final state. Hence we are safe to use F_P as the final state of $\epsilon(P)$

The set of transitions $\longrightarrow_{\epsilon(P)}$ is constructed as follow. Let $x \xrightarrow{a} x'$ be a transition of P, there are two possible cases:

a) if a is probabilistic i.e. of the form $\mathtt{flip}_{p_1,\ldots,p_n}$ and is followed by the τ_i's, then the transition

$$x \xrightarrow{\tau} p_1\delta_{x'_1} + \ldots + p_n\delta_{x'_n}$$

is in $\longrightarrow_{\epsilon(P)}$ where $x' \xrightarrow{\tau_i} x'_i$ is a transition in P.

b) else the transition $x \xrightarrow{a} x'$ is in $\longrightarrow_{\epsilon(P)}$.

We now prove that ϵ is a monotonic function from p-**Aut** to **ProbAut**.

Proposition 14. *If $P \leq Q$ then $\epsilon(P) \leq \epsilon(Q)$.*

Proof. Assume that S is a p-simulation from P to Q. Consider the exact same relation but restricted to the state space of $\epsilon(P)$ and $\epsilon(Q)$. We show that this restriction is a probabilistic simulation.

- Obviously, $(\delta_{i_P}, \delta_{i_Q}) \in \hat{S}$.
- Let $x \xrightarrow{a} \phi$ and $(x, \psi) \in \hat{S}$. Since τ transitions only occur as part of probabilistic choices, we have two possibilities:
  - $x \xrightarrow{\tau} p_1\delta_{x'_1} + \ldots + p_n\delta_{x'_n}$ is a transition of $\epsilon(P)$ and $(x, \psi) \in S$ where $\psi = \delta_y$. Since (x, y) belongs to the original S. In this case, $y \xrightarrow{\tau} p_1\delta_{y'_1} + \ldots + p_n\delta_{y'_n}$ is a transition of $\epsilon(Q)$ and each (x'_i, y'_i) belongs to the original S (Definition of p-simulation).
  - $x \xrightarrow{a} x'$ and a is an external action. Therefore there are two possibilities again, $y \xrightarrow{\tau_i} y_i \xrightarrow{a} y'$ or $y \xrightarrow{a} y'$. In both cases, we have $(x', y') \in S$.
- Conservation of final states follows easily from the fact that S is a p-simulation. □

Since our Definition (13) implies the definition of probabilistic simulation in [3], we conclude that maximal probability of doing a particular action in p-automata is increased by p-simulation. This remark provides a formal justification of our earlier example. That is, Equation (14) ensures that the maximal probability that a buyer will be satisfied when using the probabilistic vending machine is at least $1/2$ because the maximal probability of a trace containing `tea` in the automata described by

$$\mathtt{coin} \cdot \mathtt{flip} \cdot (\tau_h \cdot (\mathtt{tea} + 1) + \tau_t$$

is $1/2$.

In the proof of proposition 14, the simulation constructed is a very particular case of probabilistic simulation so it is too weak to establish certain relationships between p-automata. For instance, the automaton represented by $a_p \oplus (a_q \oplus b)$ should be equivalent to $a_{p+q-pq} \oplus b$ but Definition 11 will not provide such equality. This line of research is part of our future work where we will study proper probabilistic automata and simulations against weak concurrent Kleene algebra.

6 Algebraic Testing

In this section, we describe an algebraic treatment of *testing*. Testing is a natural ordering for processes that was studied first in [14]. The idea is to "measure" the behaviour of the process with respect to the environment. In other words, given two processes x and y and a set of test processes T, the goal is to compare the processes $x\|t$ and $y\|t$ for every $t \in T$. In our case, the set T will contain all processes.

We consider a function o from the set of terms to the set of internal expressions $I = \{x \mid x \leq 1\}$. The function $o : T_\Sigma \to I$ is defined by

$o(x) = x$ if $x \in I$	$o(st) = o(s)o(t)$
$o(a) = \tau$ for any a $\in \Sigma - I$	$o(s^*) = 1$
$o(s + t) = o(s) + o(t)$	$o(s\|t) \leq o(s)o(t)$

In the model, the function o is interpreted by substituting each external action with the internal action τ ($o(a) = \tau$ for any $a \in \Sigma - I$). Then any final state is labelled by 1 and deadlock states are labelled by 0. Inductively, we label a state that leads to some final state by 1, else it is labelled by 0. This is motivated by the fact that $x0 = 0$ for any $x \in I$ so each transition leading to *deadlock states only* will be removed. Therefore, only states labelled by 1 will remain and the transitions between them. Hence, $o(s) \neq 0$ iff the resulting automaton contains at least one state labelled by 1. In other words, $o(s) = 0$ iff x *must not terminate successfully.*

Without loss of generality (by considering automata modulo simulation), we assume that τ is the only internal action in Σ and it satisfies $\tau\tau = \tau$. This equation is valid in the concrete model.

The existence of a well-defined function o satisfying these conditions depends on our definition of simulation. That is, we can show that if $P \leq Q$ then $o(P) \leq o(Q)$ where we have abused notation by writing $o(P)$ as the application of o on the term associated to P. A detailed discussion about this can be found in the appendix under Remark 28.

Definition 15. *The may testing order is given by*

$$x \sqsubseteq_{may} y \quad \textit{iff} \quad \forall t \in T_\Sigma.\, [o(y\|t) = 0 \Rightarrow o(x\|t) = 0]\,.$$ [9]

We now provide some results about algebraic may testing. It follows from monotonicity of $\|$ with respect to $\leq$ (Proposition 3) that may ordering $\sqsubseteq_{\mathtt{may}}$ is weaker than the rooted η-simulation order.

Proposition 16. *$x \leq y$ implies $x \sqsubseteq_{may} y$.*

In fact, $\sqsubseteq_{\mathtt{may}}$ is too weak compared to $\leq$: may testing is equivalent to language equivalence. Given a term s, the language $Tr(s)$ of s is the set of finite words formed by external actions and are accepted by the automata represented by s. In other word, it is the set of finite traces in the sense of CSP which lead to final states. The precise definition of this language equivalence can be found in the appendix and so is the proof of the following proposition (Proposition 29 of the appendix).

Proposition 17. *In* **Aut**, *$\sqsubseteq_{may}$ reduces to language equivalence.*

[9] Notice $\|$ should be framed because some external actions are not synchronised. But in the setting of testing, we can also assume that all external actions are synchronised which permits to follow up all external actions present in the process.

We have shown that $\sqsubseteq_{\texttt{may}}$ is equivalent to language equivalence and hence it is weaker than our simulation order. This is also a consequence of the fact that our study of may testing is done in a qualitative way because the probabilities are found implicitly within actions. A quantitative study of probabilistic testing orders can be found in [3].

7 Case Study: Rabin's Choice Coordination

The problem of choice coordination is well known in the area of distributed systems. It usually appears in the form of processes voting for a common goal among some possibilities. Rabin has proposed a probabilistic protocol which solves the problem [15] and a sequential specification can be found in [11].

We specify the protocol in our algebra and prove that a fully concurrent specification is equivalent to a sequential one. Once this has been done, the full verification can proceed by reusing the techniques for sequential reasoning [11].

The protocol consists of a set of tourists and two places: a church C and a museum M. Each tourist has a notepad where he keeps track of an integer k. Each place has a board where tourists can read and write. We denote by L (resp. R) the value on the church board (resp. museum board).

In this section, we use $\cdot$ again for the sequential composition to make the specifications clearer.

- The church is specified as $C = (c!L)^* \cdot (c?L)$ where the channel c represents the church's door. $c!L$ means that the value of L is available to be read in the channel c and $c?L$ waits for an input which is used as value for L in the subsequent process.

  In other words, each tourist can read as many times as they want from the church board but write on it only once. Repeated writing will be considered in the specification of the protocol.

  Similarly, the museum is specified as $M = (m!R)^* \cdot (m?R)$.

- Each tourist is specified as $P(\alpha, k)$ where $\alpha \in \{c, m\}$ is the door before which the tourist currently stands and k is the actual value written on his notepad. A detailed description of P can be found in the appendix but roughly, we have

  $$P(\alpha, k) = (\alpha?K) \cdot \texttt{rabin} \cdot [\alpha := \underline{\alpha}]$$ [10]

  where $\underline{c} = m$ and $\underline{m} = c$. In other words, the tourist reads the value on the place specified by α, executes Rabin's protocol `rabin` and then goes to the other place. Notice that the process `rabin` contains the probabilistic component of Rabin's protocol. Essentially, it describes the rules that are used by each tourist to update their actual value for k with respect to the value on the board and vice versa.

  The whole specification of the protocol executed by each tourist is described by the automata of Figure 2

We are ready to specify the whole system. Assume we have two tourists P and Q (our result generalises easily to n tourists). The tourists' joint action is specified as $(P + Q)^*$. This ensures that when a tourist has started his turn by reading the board, he will not be interrupted by any other tourist until he is done and goes inside the current place or to the other place. This condition is crucial for the protocol to work properly.

The actions of the locations process are specified by $(M + C)^*$ which ensures that each tourist can be at one place at a time only — this is a physical constraint. Now, the whole system is specified by

$$\texttt{init} \cdot \Big([P(\alpha, u) + Q(\beta, v)]^*_{\{c,m\}} \| (M + C)^* \Big) \quad (16)$$

where `init` is the initialisation of the values on the boards, notepads and initial locations. Specification 16 describes the most arbitrary behaviour of the tourists compatible with visiting and interacting with the locations in the manner described above. Rabin's design of the protocol means that this behaviour is equivalent to a serialised execution where first one location is visited, followed by the other. We can write that behaviour behaviour as $[((P + Q)\|M)^*((P + Q)\|C)^*]^*$, where (for this section only) we denote the concurrency operator by $\|$ instead of $_{\{c,m\}}\|$ to make the notation lighter. The next theorem says that this more uniform execution is included in $S = [P(\alpha, u) + Q(\beta, v)]^* \| (M + C)^*$, described by Specification 16.

Theorem 18. *We have*

$$S \geq [((P + Q)\|M)^*((P + Q)\|C)^*]^*$$

The proof is a simple application of Proposition 3. Theorem 18 means S could execute all possible actions related to door M, and then those at door C, and then back to door M and so one. In fact, we can also prove the converse i.e. Proposition 18 could be strengthen to equality. But for that, we need the continuity of the operators $\cdot$ and $\|$.

Theorem 19. *In the concrete model, the specification of Rabin's protocol satisfies*

$$S = [((P + Q)\|M)^*((P + Q)\|C)^*]^*$$

The proof of this theorem depends heavily on the fact that the concurrent and sequential compositions are continuous in the the concrete model. The complete proof can be found in the appendix.

In the proof, if we stopped at the distribution over $\|$, we obtain the equivalent specification

$$S = [(P + Q)\|M + (P + Q)\|C]^*$$

which describes a simpler situation where P or Q interacts at the Museum or at the Church. This is similar to the sequential version found in [11], which can be treated by standard probabilistic invariants to complete a full probabilistic analysis of the protocol.

8 Conclusion

An algebraic account of probabilistic and concurrent system has been presented in this paper. The idea was to combine probabilistic and concurrent Kleene algebra. A soundness result with respect to automata and rooted η-simulation has been provided. The concrete model ensures not only the consistency of the axioms but provides also a semantic space for systems exhibiting probabilistic, nondeterministic and concurrent behaviour. We also showed that the model has

[10] Any action written within square brackets will denote internal action (see appendix for the detailed specification).

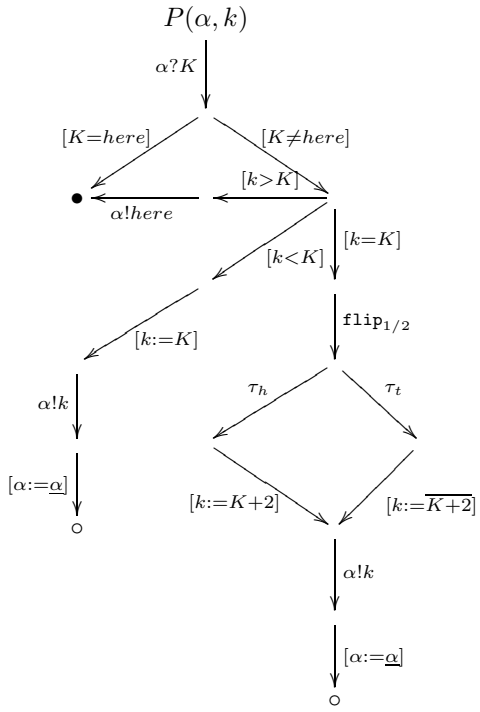


Figure 2: p-automaton that describes the protocol $P(\alpha, k)$ executed by each tourist.

stronger properties than just the algebraic axiomatisation. For instance, sequential and concurrent compositions are both continuous in the case of finite automata.

We provided some applications of the framework. An algebraic account of may testing has been discussed in Section 6. It was shown that may ordering reduces to language equivalence.

We also provided a case study of Rabin's solution to the choice coordination problem. A concurrent specification was provided and it was shown to be structurally equivalent to the sequential one given in [11].

Though the algebra was proven to be powerful enough to derive non-trivial properties for concrete protocols, the concrete model still needs to be refined. For instance, the inclusion of tests is important especially for the construction of probabilistic choices. Tests need to be introduced carefully because their algebraic characterisation are subtle due to presence of probability. We also need to improve and refine the manipulation of quantitative properties in the model as part of our future work.

Finally, it is customary to motivate automated support for algebraic approaches. The axioms system for weak concurrent Kleene algebra is entirely first-order, therefore proof automation is supported and automatised version of our algebraic proofs can be found in our repository.

References

[1] E. Cohen. Weak Kleene algebra is sound and (possibly) complete for simulation. *CoRR*, abs/0910.1028, 2009.

[2] J. H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, Mathematics series, 1971.

[3] Y. Deng and R. Van Glabbeek. Characterising testing preorders for finite probabilistic processes. In *In LICS07: Proceedings of the 22nd Annual IEEE Symposium on Logic in Computer Science. IEEE Computer Society Press, Los Alamitos, CA*, pages 313–325, 2007.

[4] C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21:666–677, August 1978.

[5] C. A. R. Hoare, B. Möller, and I. Struth, G.and Wehrman. Concurrent Kleene algebra. In *Proceedings of the 20th International Conference on Concurrency Theory*, CONCUR 2009, pages 399–414, Berlin, Heidelberg, 2009. Springer-Verlag.

[6] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Infor. and Comput.*, 110(2):366–390, May 1994.

[7] D. Kozen. On Hoare logic and Kleene algebra with tests. *Trans. Computational Logic*, 1(1):60–76, July 2000.

[8] D. Kozen and M. C. Patron. Certification of compiler optimizations using Kleene algebra with tests. In John Lloyd, Veronica Dahl, Ulrich Furbach, Manfred Kerber, Kung-Kiu Lau, Catuscia Palamidessi, Luis Moniz Pereira, Yehoshua Sagiv, and Peter J. Stuckey, editors, *Proc. 1st Int. Conf. Computational Logic (CL2000)*, volume 1861 of *LNAI*, pages 568–582, London, July 2000. Springer-Verlag.

[9] A. McIver, T. M. Rabehaja, and G. Struth. On probabilistic Kleene algebras, automata and simulations. In *Proceedings of the 12th international conference on Relational and algebraic methods in computer science*, RAMICS'11, pages 264–279, Berlin, Heidelberg, 2011. Springer-Verlag.

[10] A. K. McIver, E. Cohen, and C. C. Morgan. Using probabilistic Kleene algebra for protocol ver-

ification. In *In Relmics/AKA 2006, volume 4136 of LNCS*. Springer Verlag.

[11] A. K. McIver and C. C. Morgan. *Abstraction, Refinement And Proof For Probabilistic Systems (Monographs in Computer Science)*. SpringerVerlag, 2004.

[12] A. K. McIver and T. Weber. Towards automated proof support for probabilistic distributed systems. In *In Proceedings of Logic for Programming and Automated Reasoning, volume 3835 of LNAI*, pages 534–548. Springer, 2005.

[13] R. Milner. An algebraic definition of simulation between programs. Technical report, Stanford, CA, USA, 1971.

[14] R. De Nicola and M. Hennessy. Testing equivalence for processes. In *Proceedings of the 10th Colloquium on Automata, Languages and Programming*, pages 548–560, London, UK, 1983. Springer-Verlag.

[15] M. O. Rabin. The choice coordination problem. *Acta Inf.*, 17:121–134, 1982.

[16] R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. In *Nordic Journal of Computing*, pages 481–496. Springer, 1994.

[17] R. G. van Glabbeek. The linear time-branching time spectrum (extended abstract). In J. C. M. Baeten and J. W. Klop, editors, *CONCUR 1990*, volume 458 of *LNCS*, pages 278–297. Springer, 1990.

Appendix

The following proofs, diagrams, remarks and other results are only included to add further clarification of the contents of the present paper. It is left to the discression of the reviewers to choose whether they will read these proofs or not.

A Diagrams, Theorems and Proofs

Diagram of the Operators: The construction are done inductively from $0, 1$ and elements of the alphabet Σ.

- **Deadlock:** 0.
  This is the automaton that has only one state, no transition and no final state.
- **Skip:** 1
  This is the automaton $\circ$ which has only one state which is both initial and final and has no transition.
- **Single action:**
  The automaton associated to $a \in \Sigma$ is $i \xrightarrow{a} \circ$ where i is the initial state and $\circ$ is a final state.
- **Addition:** $P + Q$.
  This is constructed by identifying the initial states of P and Q. This construction is allowed because of the initiality condition (Figure 3).
- **Multiplication:** PQ.
  This is constructed by identifying each final state of P with the initial state of Q (Figure 4).
- **Concurrency:** $P_A \| Q$
  This is constructed as a sub-automaton of the Cartesian product of P and Q following CSP [4]. Assuming $a \in A$ and $b, d \notin A$, the concurrent composition $P_A \| Q$ is inductively constructed as in Figure 5

  Notice that $A \subseteq \Sigma$ is a set of synchronised action and does not contain any (strictly) probabilistic actions such as $\texttt{flip}(p)$, for $p \in]0, 1[$.
- **Kleene star:** P^*
  This is the result of repeating P allowing a successful termination after each (possibly empty) full execution of P. In the diagram of Figure 6, we just picture one transition from the initial state and one final state. The construction needs to be performedfor each initial transition and final state. Notice the initial state of P^* is a final state too.

Proposition 20. *These operations are well defined on* **Aut** *that is if* $P, Q \in$ **Aut** *then* $P + Q, PQ, P_A \| Q$ *and* P^* *are elements of* **Aut**.

Proof. The proof is by induction on the structure of the automata P and Q. For the base case, it is obvious that $0, 1$ and $i \xrightarrow{a} \circ$ satisfy the reachability and initiality conditions.

Let $P, Q \in$ **Aut**. It is easy to see from the diagrams that $P + Q, P \| Q$ and P^* belongs to **Aut** too. PQ satisfies the initiality condition because the initial state is i_P. For reachability, let $x \in Q$. Then x is reachable from i_Q which in turn is reachable from i_P by the definition of sequential composition. □

Proposition 21. *The following statements hold.*

1. *The relational composition of two rooted η-simulations is again a rooted η-simulation. That is, if S, T are rooted η-simulations then $S \circ T$ is also a rooted η-simulation, where $\circ$ denotes relational composition.*
2. *The simulation relation $\leq$ is a preorder on* **Aut**.

Proof. 1. Let $S : P \to Q$ and $T : Q \to R$ be simulations and let us show that $ST : P \to R$ is a simulation.

- Evidently, $(i, i) \in ST$.
- Let $(x, z) \in ST$ and $x \xrightarrow{a} x'$. By definition of the relational composition there exists $y \in Q$ such that $(x, y) \in S$ and $(y, z) \in T$.
  a) if a is internal, there exists $y' \in Q$ such that $y \Rightarrow y'$ and $(x', y') \in S$. Since $y \Rightarrow y'$ consists of a sequence of finite internal transition, there exists $z' \in T$ such that $(y', z') \in T$ and $z \Rightarrow z'$. Hence $(x', z') \in ST$ and $z \Rightarrow z'$.
  b) If a is external, there exists $y_1, y' \in Q$ such that $y \Rightarrow y_1 \xrightarrow{a} y'$ and $(x, y_1) \in S$ and $(x', y') \in S$. Since $(y, z) \in T$ and $y \Rightarrow y_1$, there exists $z_1 \in R$ such that $(y_1, z_1) \in T$ and $z \Rightarrow z_1$. Again, since T is a simulation and $y_1 \xrightarrow{a} y'$, there exists $z_2, z' \in R$ such that $z_1 \Rightarrow z_2 \xrightarrow{a} z'$ and $(y_1, z_2) \in T$ and $(y', z') \in T$. Hence, by transitivity of $\Rightarrow$, we have $z \Rightarrow z_2 \xrightarrow{a} z'$ and $(x, z_2) \in ST$ and $(x', z') \in ST$.
- Let $(x, z) \in ST$ and $x \in F_P$, there exists $y \in Q$ such that $(x, y) \in S$ and $(y, z) \in T$. So $y \in F_Q$ and hence $z \in F_R$.
- Let $(i, z) \in ST$, there exists $y \in Q$ such that $(i, y) \in S$ and $(y, z) \in T$. So $y = i$ and hence $z = i$.

2. For reflexivity, the identity relation is a rooted η-simulation and transitivity follows from 1. □

Proposition 22. *The equivalence relation $\equiv$ is a congruence with respect to $+$ and $P \leq Q$ iff $P + Q \equiv Q$.*

Proof. Let $S : P \to Q$ and $S' : P' \to Q'$ be routed η-simulations. We show that $S \cup S' : P + P' \to Q + Q'$ is again a routed η-simulation.

- Since initial states are identified in the construction of $+$, we have $(i_{P+P'}, i_{Q+Q'}) = (i_P, i_Q) \in S \cup S'$.
- Let $(x, y) \in S \cup S'$ and $x \xrightarrow{a} x'$ be a transition of P (the case where this transition belongs to P' is dealt with the exact same way). We have two cases:
  1. if $(x, y) \in S$, then either a is internal and hence $(x', y) \in S$ (so in $S \cup S'$ too) or there exists $y_1, y' \in Q$ such that $y \Rightarrow y_1 \xrightarrow{a} y'$ is a path in Q and $(x, y_1) \in S$ and $(x', y') \in S$. By definition of $+$, $y \Rightarrow y_1 \to y'$ is again a path in $Q + Q'$ such that $(x, y_1) \in S \cup S'$ and $(x', y') \in S \cup S'$.

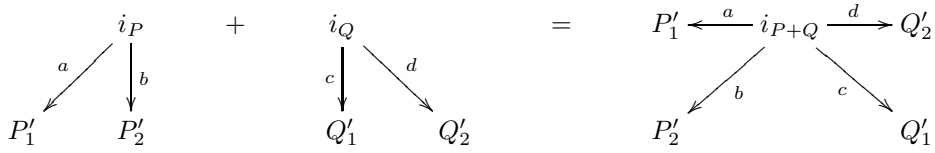


Figure 3: Automaton for $P + Q$.

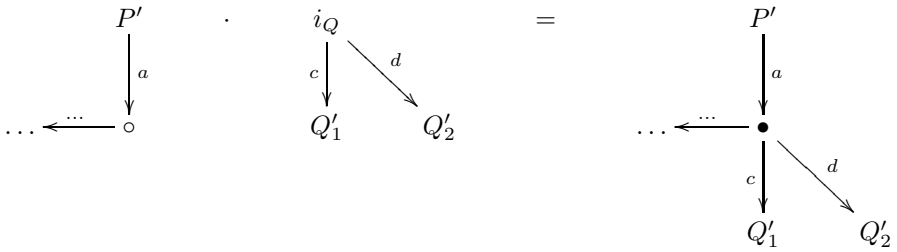


Figure 4: Automaton for PQ. The symbol $\circ$ denotes a final state and the symbol $\bullet$ is final if and only if i_Q is final in Q. Notice that this construction is done for each final state of P.

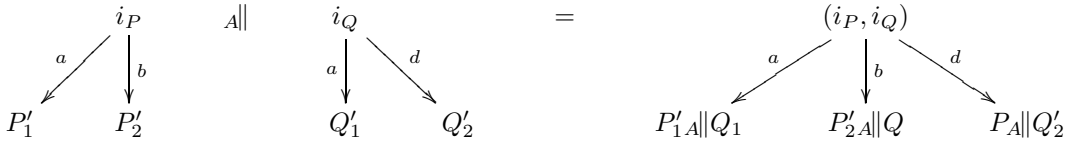


Figure 5: Automaton for $P_A\|Q$. The action a has been synchronised and b, d were interleaved. Notice that b or d could be internal. The initial state of the automata is the pair (i_P, i_Q) and the final states are the elements of $F_P \times F_Q$.

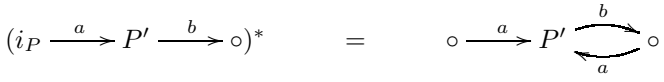


Figure 6: Automaton for P^*.

2. if $(x,y) \in S'$, then $x = i_P$ because $x \xrightarrow{a} x'$ is assumed to be a transition in P. Since the initial states are merged, $(i_{P'}, y) \in S'$ and therefore $y = i_{Q'} = i_{Q+Q'} = i_Q$. Therefore $(x,y) \in S$ and we are back to Case 1.

- Let $(x,y) \in S \cup S'$ and $x \in F_P$ (the case $x \in F_{P'}$ is similar). We have two cases again, $(x,y) \in S$ and $y \in F_{P'}$. Or $(x,y) \in S'$ and then $x \in P \cap P'$. Hence $x = i$ and we are back to the first case again.

- $S \cup S'$ is rooted because S and S' are both rooted.

Now assume $P \leq Q$. Then $P + Q \leq Q + Q \equiv Q$ follows from the fact that $\leq$ is a congruence and the idempotence of $+$ in Proposition 8. Moreover, since $id_Q : Q \to Q$ is a simulation, we have $P + Q \equiv Q$. Conversely, assume $P + Q \equiv Q$, since $id_P : P \to P + Q$ is a simulation we have $P \leq Q$ by transitivity of $\leq$. □

Proposition 23. $(\mathbf{Aut}, +, \cdot, *, 0, 1)$ *is a weak probabilistic Kleene algebra.*

Proof. Associativity and commutativity of $+$ and $0 + x = x$ follows easily from the fact that $+$ is base on $\cup$.

- Idempotence of $+$: since the union is made disjoint, we assume P_c is a copy of P where every states is indexed by c. Then $id_P : P \to P + P_c$ is a simulation and $\{(y,x) \mid y = x \text{ or } y = x_c\}$ is a simulation from $P + P_c$ to P.

- Associativity of $\cdot$: associativity follows from the same proof found in [9] because identity relations are simulation and our multiplication here is exactly the ε-free version of the multiplication there.

- 1 is neutral for $\cdot$: it follows easily from the construction that $1P = P$ and $P1 = P$.

- Subdistributivity 3: to show that $PQ + PR \leq P(Q+R)$, it suffices to show that $PR \leq P(Q+R)$ and derive the result from idempotence of $+$. Remind that id_P and id_Q are simulation so it suffices to show that $id_P \cup id_Q : PQ \to P(Q + R)$ is again a simulation. Obviously, $(i,i) \in S$ and it is rooted and conserves final states. Moreover $\longrightarrow_{P(Q+R)} \supseteq \longrightarrow_{PQ}$. Hence $id_P \cup id_Q$ is a simulation.

- Right distributivity 5: let P_c be a disjoint copy of P, then the relation

$$S = \{(x,y) \mid y = x \text{ or } y = x_c\}$$

from $(Q + R)P$ to $QP_c + RP$ is rooted and preserves final states. Let $(x,y) \in S$ and $x \xrightarrow{a} x' \in \longrightarrow_{(Q+R)P}$. Remind that

$$\longrightarrow_{(Q+R)P} = \longrightarrow_Q \cup \longrightarrow_R \cup \longrightarrow_P - \{i \xrightarrow{a} z \in \longrightarrow_P\}$$
$$\cup \{z \xrightarrow{a} z' \mid z \in F_Q \cup F_R \text{ and } i \xrightarrow{a} z' \in \longrightarrow_P\}$$

If the transition belongs to the first three sets then we are done, else we can assume $x \in F_Q$ and $i \xrightarrow{a} x' \in \longrightarrow_P$ i.e. $y = x$ and $x' \in P$. We have

$$\longrightarrow_{QP_c+RP} \supseteq \{z \xrightarrow{a} z'_c \mid z \in F_Q \text{ and } i \xrightarrow{a} z'_c \in \longrightarrow_{P_c}\}$$

so $x \xrightarrow{a} x'_c \in \longrightarrow_{QP+RP}$ and $(x', x'_c) \in S$. Similarly, we can prove that if $(x,y) \in S$ and $y \xrightarrow{a} y'$ then there exists x' such that $(x'y') \in S$ and $x \xrightarrow{a} x'$. Hence S is a bisimulation.

- Left unfold 2: Let x_* be a state in P^* and x the corresponding state in the unfolded version $(1 + PP^*)$ i.e. x is considered as a state of P. The rooted version of relation $S = id_{P^*} \cup \{(x_*, x)\}$ is a rooted η-bisimulation from P^* to $1 + PP^*$.

- Left induction 3: as in (10), the proof is again similar to [9] because rooted η-simulation are stable by union.

□

Proposition 24. $(\mathbf{Aut}, +, \cdot, {}_A\|, 1)$ *satisfies equations (9- 13) modulo rooted η-simulation equivalence for any set of synchronisable actions $A \subseteq \Sigma$ (i.e. no probabilistic actions).*

Proof. $1_A\|1 = 1$ follows directly from the definition of ${}_A\|$ and the simulation used for the commutativity is $\{((x,y),(y,x)) \mid x \in P \text{ and } y \in Q\}$.

(9) For associativity, we show that if $(x,(y,z)) \xrightarrow{a} (x',(y',z')) \in \longrightarrow_{P_A\|(Q_A\|R)}$ then $((x,y),z) \xrightarrow{a} ((x',y'),z') \in \longrightarrow_{(P_A\|Q)_A\|R}$.

- If $a \notin A$, then
  - $x \xrightarrow{a} x'$ and $y = y', z = z'$. So $(x,y) \xrightarrow{a} (x',y) \in \longrightarrow_{P_A\|Q}$ and hence $((x,y),z) \xrightarrow{a} ((x',y),z) \in \longrightarrow_{(P_A\|Q)_A\|R}$ because $a \notin A$.
  - or $x = x'$ and $(y,z) \xrightarrow{a} (y',z')$. Since $a \notin A$:
    - $y \xrightarrow{a} y'$ and $z = z'$, hence $((x,y),z) \xrightarrow{a} ((x,y'),z) \in \longrightarrow_{(P_A\|Q)_A\|R}$,
    - or $y = y'$ and $z \xrightarrow{a} z'$ and hence $((x,y),z) \xrightarrow{a} ((x,y),z') \in \longrightarrow_{(P_A\|Q)_A\|R}$.
- If $a \in A$, then $x \xrightarrow{a} x'$ and $(y,z) \xrightarrow{a} (y',z')$. Since a is again synchronised in $Q_A\|R$, $y \xrightarrow{a} y'$ and $z \xrightarrow{a} z'$. So $(x,y) \xrightarrow{a} (x',y') \in \longrightarrow_{P_A\|Q}$ and hence $((x,y),z) \xrightarrow{a} ((x',y'),z') \in \longrightarrow_{(P_A\|Q)_A\|R}$.

Since ${}_A\|$ is commutative, we deduce that $\longrightarrow_{(P_A\|Q)_A\|R} = \longrightarrow_{P_A\|(Q_A\|R)}$ so the identity relations could again be used for the simulation.

(12) To prove monotonicity, we consider the relation $S : P_A\|Q + P_{cA}\|R \to P_A\|(Q + R)$ as in the case of multiplication i.e. $S = \{((x,y),(x,y)),((x_c,y),(x,y)) \mid x \in P \wedge y \in Q \cup R\}$ and x_c is the copy of the state $x \in P$ in P_C. Let $((x_c,y),(x,y)) \in S$ (the case $((x,y),(x,y)) \in S$ is easier and can be handled in the same way) and $(x_c,y) \xrightarrow{a} (x'_c,y') \in \longrightarrow_{P_A\|Q+P_A\|R}$. By definition of $+$, that transition belongs to $\longrightarrow_{P_A\|Q}$ or $\longrightarrow_{P_A\|R}$. Since the first component is a copy of x, we have $(x_c,y) \xrightarrow{a} (x'_c,y') \in \longrightarrow_{P_A\|R}$ that is $y, y' \in R$.

– if $a \notin A$, then
  * $x_c \xrightarrow{a} x'_c$ and $y = y'$, so $x \xrightarrow{a} x' \in \longrightarrow_P$ and hence $(x,y) \xrightarrow{a} (x',y) \in \longrightarrow_{P_A\|(Q+R)}$ and $((x'_c,y),(x',y)) \in S$ by definition of S.
  * or $x_c = x'_c$ and $y \xrightarrow{a} y'$, so $(x,y) \xrightarrow{a} (x,y') \in \longrightarrow_{P_A\|(Q+R)}$ and $((x,y'),(x_c,y')) \in S$.
– if $a \in A$, then $x_c \xrightarrow{a} x'_c$ and $y \xrightarrow{a} y \in \longrightarrow'_R$. So $x \xrightarrow{a} x' \in \longrightarrow_P$ and hence $(x,y) \xrightarrow{a} (x',y') \in \longrightarrow_{P_A\|(Q+R)}$ and $((x'_c,y'),(x',y')) \in S$.

(13) Firstly notice that the set of states of $(P\|Q)(P'\|Q')$ (where the frame A of the concurrency operator is left implicit) is a subset of $(P \times Q) \cup (P' \times Q')$ which is in turn a subset of $(P \cup P') \times (Q \cup Q')$. Hence we consider the injection id of the former set to the later one and the relation defined in Figure A We show that S is a simulation in our sense.

- Since $(i,i) = i$ is related to itself. In particular, S is rooted because $x' \neq i$ in the second set in the definition of S (resp. for the third set) and HCI.
- Let $(x,y) \in (P\|Q)(P'\|Q')$ such that $(x,y) \xrightarrow{a} (x',y')$. We have the following cases.
  * The transition is in $\longrightarrow_{P\|Q}$, in which case $(x,y),(x',y') \in P \times Q$.
    · if $a \notin A$ then $x \xrightarrow{a} x' \in \longrightarrow_P$ and $y = y'$ or $x = x'$ and $y \xrightarrow{a} y' \in \longrightarrow_Q$. By definition of the sequential composition again, these transitions belong to $\longrightarrow_{PP'}$ or $\longrightarrow_{QQ'}$ respectively. Hence $(x,y) \xrightarrow{a} (x',y') \in \longrightarrow_{PP'\|QQ'}$.
    · if $a \in A$ then $x \xrightarrow{a} x' \in \longrightarrow_P$ and $y \xrightarrow{a} y' \in \longrightarrow_Q$. As in the previous case, the considered transition exists in $PP'\|QQ'$
  * The transition is in $\longrightarrow_{P'\|Q'-\{(i,i)\}}$. This case is similar to the previous one because because $x \neq i$ and $y \neq i$ as states of P' and Q' respectively.
  * It is a linking transition i.e. $(x,y) \in F_{P\|Q}$ and $(i,i) \xrightarrow{a} (x',y') \in \longrightarrow_{P'\|Q'}$. Then $x \in F_P$ and $y \in F_Q$ and we have two cases:
    · if $a \notin A$, then $i \xrightarrow{a} x' \in \longrightarrow_{P'}$ and $y' = i$ or $x' = i$ and $i \xrightarrow{a} y' \in \longrightarrow_{Q'}$. In the first case, the definition of S implies that $((x',y'),(x',y)) \in S$ and since $a \notin A$, we have $(x,y) \xrightarrow{a} (x',y) \in \longrightarrow_{PP'\|QQ'}$. Similarly for the other case.
    · if $a \in A$, then $i \xrightarrow{a} x' \in \longrightarrow_{P'}$ and $i \xrightarrow{a} y' \in \longrightarrow_{Q'}$. Then $x \xrightarrow{a} x' \in \longrightarrow_{PP'}$ and $y \xrightarrow{a} y' \in \longrightarrow_{QQ'}$. Hence $(x,y) \xrightarrow{a} (x',y') \in \longrightarrow_{PP'\|QQ'}$.
  * Let $((x',i),(x',y)) \in S$ as in the above definition of S and $(x',i) \xrightarrow{a} (x'',y'') \in \longrightarrow_{P'\|Q'}$.
    · If $a \notin A$, then $x' \xrightarrow{a} x'' \in \longrightarrow_{P'}$ and $y'' = i$ or $x' = x''$ and $i \xrightarrow{a} y'' \in \longrightarrow_{Q'}$. In the first case, $(x',y) \xrightarrow{a} (x'',y) \in \longrightarrow_{PP'\|QQ'}$ because $a \notin A$ and $(x'',y) \in S$ because $y'' = i$. In the second case, $y \xrightarrow{a} y'' \in \longrightarrow_{QQ'}$ and hence $(x',y) \xrightarrow{a} (x'',y'') \in \longrightarrow_{PP'\|QQ'}$ because $x' = x''$.
    · If $a \in A$, then $x' \xrightarrow{a} x'' \in \longrightarrow_{P'}$ and $i \xrightarrow{a} y'' \in \longrightarrow_{Q'}$. By definition of sequential composition, $y \xrightarrow{a} y'' \in \longrightarrow QQ'$ and since $a \in A$, $(x',y) \xrightarrow{a} (x'',y'') \in \longrightarrow_{PP'\|QQ'}$.
  * The case $((i,y'),(x,y')) \in S$ is similar.
- Let $((x,y),(u,v)) \in S$ such that $(x,y) \in F_{(P\|Q)(P'\|Q')}$. That is,

  $$x \in (F_{P'} - \{i\}) \cup o_{P'\|Q'}((i,i))F_P \subseteq (F_{P'} - \{i\}) \cup o_{P'}(i)F_P$$

  and similarly for y. Hence, if tuple belongs to id then we are done. Assume $i \xrightarrow{a} x \in \longrightarrow_{P'}$ and $y = i$ (the other case is proved in exactly the same way), then $u = x$ and $v \in F_Q$. Since (x,i) is a final state, we have $F_{QQ'} = (F_{Q'} - \{i\}) \cup F_Q$ and $x' \in F_{P'} - \{i\}$. Hence $(u,v) \in F_{PP'} \times F_{QQ'}$.

Finally, since simulation preserves reachability, the reachable part of $(P\|Q)(P'\|Q')$ is simulated by the reachable part of $PP'\|QQ'$.

□

Proposition 25. *The sequential composition is (conditionally) continuous from the left and the right in* $\mathbf{Aut}_f$. *That is, if* $(P_i)_i$ *is a* $\leq$*-directed set of finite automata with limit* P *then* $\sup_i P_i K = PK$ *and* $\sup_i KP_i = KP$.

We denote $\mathbf{Aut}_f$ the set of finite automata satisfying the reachability and initiality conditions. $\mathbf{Aut}_f$ is a subalgebra of $\mathbf{Aut}$. The proof is similar to our proof in [9]. The only difference is from the manipulation of 0 (because we do not have $x0 = 0$ in this setting) and hence Proposition 25 is a generalised version of the continuity in [9].

Proof. We first define a notion of residuation on $\mathbf{Aut}_f$. For automata P and Q we define the automaton P/Q with initial state $i_{P/Q} = i_P$, final states $F_{P/Q} = \{x \in P \mid Q \leq P_x\}$, where P_x is constructed from P by making its initial state into x. We make the resulting automaton reachable by discarding all states not reachable from x. Notice that P_x does not necessarily satisfy HCI. In this case, we unfold each transition from x once and isolate x but keeping a disjoint copy of it to make sure that the resulting automata is bisimulation equivalent to the non-rooted version.

We now show that $RQ \leq P$ iff $R \leq P/Q$. Assume S is a simulation from RQ to P. That means S generates a simulation from Q to P_x for some x. It follows from the definition of P/Q that S generates a simulation from R to P/Q, since the state x become

$$S = id \;\cup\; \{((x',i),(x',y)) \mid y \in F_Q \text{ and } i \xrightarrow{a} x' \in \longrightarrow_{P'} \text{ and } i \in Q'\}$$
$$\cup\; \{((i,y'),(x,y')) \mid x \in F_P \text{ and } i \xrightarrow{a} y' \in \longrightarrow_{Q'} \text{ and } i \in P'\}$$

Figure 7: Construction of the simulation to prove the interchange law.

final state of G/H and they are images of the final states of K under the simulation generated by S.

For the converse direction, suppose that S is a simulation from R to P/Q. By Theorem 8, multiplication is isotone, hence $RQ \leq (P/Q)Q$, and it remains to show that $(P/Q)Q \leq P$.

First, if $F_{P/Q}$ is empty and then R has no final state either and $RQ = R$ by definition of sequential composition. Hence $RQ = R \leq P/Q \leq P$.

Assume $F_{P/Q}$ is not empty and let S' be a simulation from RQ to $(P/Q)Q$. By construction of P/Q, we know that there exists a simulation S_x from Q to P_x for all final states $x \in F_{P/Q}$. Moreover, there is a relation $T : P/Q \to P$ satisfying all properties of simulation except the final state property, namely a restriction of the identity relation id_P. We can show that $T' = (\cup_x S_x) \cup T$ is indeed a simulation from $(P/Q)Q$ to P and $S' \circ T'$ is a simulation from RQ to P.

It then follows from general properties of Galois connections that $(\cdot H)$ is (conditionally) completely additive, hence right continuous.

It remains to show left continuity. Let $(Q_i)_i$ be a directed set of automata such that $\sup_i Q_i = Q$ and let P be any automaton. Then $\sup_i(PQ_i) \leq PQ$ because multiplication is monotone and it remains to show $PQ \leq \sup_i(PQ_i)$. Let us assume that $\sup_i(PQ_i) \leq R$. We will show that $PQ \leq R$.

By definition of supremum, $PQ_i \leq R$ for all i, hence there is a set of states $X_i = \{x \in R \mid Q_i \leq R_x\}$, that is, the set of all those states in R from which Q_i is simulated. Obviously, $X_i \subseteq X_j$ if $Q_j \leq Q_i$ in the directed collection. But since $R \in \mathbf{Aut}_f$ has only finitely many states, there must be a minimal set X in that directed set such that all Q_i are simulated by R_x for some $x \in X$. Therefore $Q = \sup_i Q_i \leq R_x$ for all $x \in X$. There exists a simulation $S_X : PQ_i \to R$ for some i such that the residual automaton R/Q_i has precisely X as its set of final states. We can thus take the union of S_X restricted to P with all simulations yielding $Q \leq R_x$ for all $x \in X$ and verify that this is indeed a simulation of PQ to R. □

We denote $L(P) = \{t \mid t \text{ is a tree and } t \leq P\}$ the tree language associated to P. We have

Lemma 26. *$P \leq Q$ iff $L(P) \subseteq L(Q)$.*

A specialized version of this theorem could be found in [9]. In this paper, we prove it for our rooted η-simulation.

Proof. By transitivity of simulation, we have $P \leq Q$ implies $L(P) \subseteq L(Q)$ so it suffices to show the converse.

Let $L(P) \subseteq L(Q)$ and consider the relation $S : P \to Q$ such that $(x,y) \in S$ iff $L(P_x) \subseteq L(Q_y)$, where P_x is the automata constructed from P with initial state x as in the previous proof. We now show that the rooted version of S is a simulation.

- Since $L(P) \subseteq L(Q)$, we have $(i_P, i_Q) \in S$.
- Let $(x,y) \in S$ and $x \in F_P$, then $1 \in L(P_x) \subseteq L(Q_y)$. Hence $y \in F_Q$.
- Let $(x,y) \in S$, $L(P_x) \subseteq L(Q_y)$ and $x \xrightarrow{a} x'$ be a transition of P. There are two cases:
  a) a is internal: for any tree t, $at \leq t$. Hence $L(P_{x'}) \subseteq L(Q_y)$ i.e. $(x',y) \in S$.
  b) a is external: assume for a contradiction that for each $y'_i \in Q$ such that $y \Rightarrow y_1 \xrightarrow{a} y'_i$, there exists $t_i \in L(P_{x'})$ such that $t_i \notin L(Q_{y'_i})$. Since $Q \in \mathbf{Aut}_f$, there are only finitely many such y'_i. By definition of η-simulation, $a(\sum_i t_i) \in L(P_x)$ and from it follows from the hypothesis that $a(\sum_i t_i) \in L(Q_y)$ i.e. $a(\sum_i t_i) \leq Q_y$. It follows from the definition of η-simulation that there exists y'_j such that $y \Rightarrow y_1 \xrightarrow{a} y'_j$ and $\sum_i t_i \leq y'_j$ which implies $t_j \leq \sum_i t_i \leq y_j$, a contradiction.
- Making the relation S rooted does not affect the well-definedness of S as a simulation because the automata P, Q are rooted.

□

Proposition 27. *The concurrency operator $\|$ is (conditionally) continuous in $\mathbf{Aut}_f$.*

Proof. We need to show that for any $\leq$-directed sequence $(Q_i)_i \subseteq \mathbf{Aut}_f$ such that $\sup_i Q_i = Q$, we have $\sup_i(P\|Q_i) = P\|Q$, where the frame is left implicit.

Firstly, we show that

$$L(P\|Q) =\downarrow (L(P)\|L(Q)) =\downarrow \{t\|t' \mid t \in L(P) \wedge t' \in L(Q)\}$$

where $\downarrow X$ is the down closure of X. Since $\|$ is monotone, $t\|t' \in L(P\|Q)$. Conversely, let $t \in L(P_A\|Q)$. By unfolding P and Q up to the depth of t, we can find two tree t_P, t_Q such that $t \leq t_P\|t_Q$ and hence $t \in\downarrow (L(P)\|L(Q))$.

Secondly, we have

$$\begin{aligned} L(P\|Q) &= \downarrow \{t\|t' \mid t \in P \wedge t' \in \cup_i L(Q_i)\} \\ &= \downarrow \cup_i \{t\|t' \mid t \in P \wedge t' \in L(Q_i)\} \\ &= \cup_i \downarrow \{t\|t' \mid t \in P \wedge t' \in L(Q_i)\} \\ &= \cup_i L(P\|Q_i) \end{aligned}$$

and directedness ensures that $L(Q) = \cup_i L(Q_i)$ [11]. Therefore, Lemma 26 ensures that $P\|Q = \sup_i(P\|Q_i)$. □

Remark 28. *The following remarks ensures the existence of an o function satisfying the properties listed in Section 6*

1. *The axiom $\tau\tau = \tau$ ensures that $o(x) \in \{0, \tau, 1\}$ for any $x \in T_\Sigma$. If $o(x) = 0$ then x will never terminate successfully. If $o(x) = 1$, then x may terminate successfully without the execution of any*

[11] $\cup_i L_i \subseteq Q$ is obvious and the converse could be proven by showing that if $t \in L(Q)$ and $t \notin L(Q_i)$ for every i, then there exists Q' constructed from Q "minus" some part of t such that $Q_i \leq Q' < Q$ for any i.

action [12]*, and if* $o(x) = \tau$ *then* x *may terminate successfully after the execution of some action.*

2. *The interpretation of* o *in the concrete model respects simulation. In fact, let* P, Q *be the automata representing some terms in* T_Σ *and* $S : P \to Q$ *be a simulation. After replacing each action in* P, Q *by* τ*,* S *remains a simulation by Propriety (a) of Definition 5. Therefore*
   - *if* $o(P) = 1$ *then the initial state of* P *is final and so is the initial state of* Q*,*
   - *if* $o(P) = \tau$ *then the initial state of* P *leads to some final state and so is the initial state of* Q *i.e.* $1 \leq o(Q)$*,*
   - *if* $o(P) = 0$ *then we are done,*

   and in all three cases $o(P) \leq o(Q)$*. Hence, it is safe to assume that* o *is well defined on* T_Σ *modulo the axioms of weak concurrent Kleene algebra. In particular,* o *is monotonic with respect to the restriction of the natural order of the algebra on* I*.*

3. *The last property* $o(x\|y) \leq o(x)o(y)$ *is in general a strict inequality. For instance, if* a, b *are synchronised actions then* $o(a\|b) = o(0) = 0$ *but* $o(a)o(b) = \tau\tau = \tau$*.*

Proposition 29. *In* **Aut**, $\sqsubseteq_{may}$ *reduces to language equivalence.*

Remind that we assume there is only one non-trivial internal action, namely τ, and it satisfies $\tau\tau = \tau$.

Proof. Firstly, the language of the automata associated to x is given by

$$Tr(x) = \{t \mid t \text{ is linear, loop-free, has only } \tau \text{ as non-synchronised action and } t \leq_\tau x\}$$

where $x \leq_\tau y$ if there is a simulation between the automata represented by x and y such that all non-synchronised actions are replaced by τ. This ensures for instance that $Tr(\texttt{flip}) = \{\tau\}$.

Remind that $Tr(x\|y) = Tr(x) \cap Tr(y)$ [13] because elements of $Tr(x)$ are of the form $w\tau$ or w (modulo the equivalence from $\leq_\tau$) where w is a word formed of synchronised actions only.

For the direct implication, assume $x \sqsubseteq_{\texttt{may}} y$ and let $t \in Tr(x)$. Then $o(x\|t) \neq 0$ and since $x\|t \leq y\|t$, we have $o(y\|t) \neq 0$. Since t has synchronised actions only (or possibly ends with τ) and $o(y\|t) \neq 0$, then $t \in Tr(y)$ that is $Tr(x) \subseteq Tr(y)$.

Conversely, let $Tr(x) \subseteq Tr(y)$ and $z \in T_\Sigma$. $Tr(x\|z) = Tr(x) \cap Tr(z) \subseteq Tr(y) \cap Tr(z) = Tr(y\|z)$. So if $o(y\|z) = 0$ then $y\|z$ has no final state and hence $Tr(y\|z) = \{0\}$. Hence $Tr(x\|z) = \{0\}$ i.e. $x\|z$ has no final state that is $o(x\|z) = 0$. □

B Specification of Rabin's Protocol.

Remind that $P(\alpha, k)$ is the specification of a tourist in from of the door $\alpha \in \{m, c\}$ and has k written on his notepad (Figure 8).

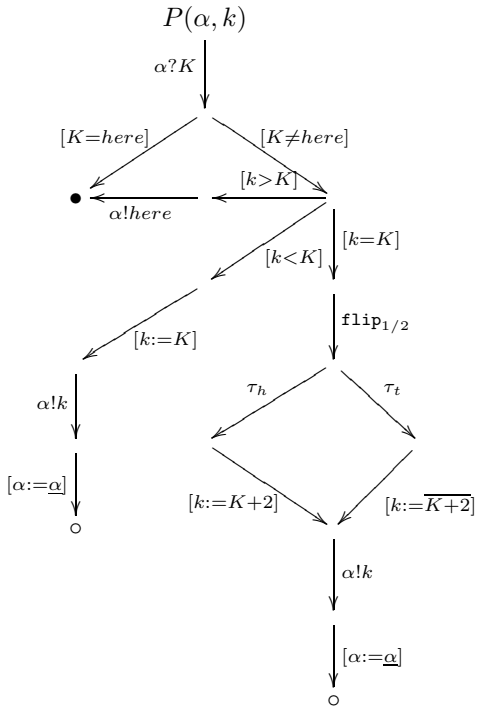


Figure 8: Interpretation of $P(\alpha, k)$ in term of automata with imiplicit probability.

Any action of the form $[a]$ are considered internal. The symbol $\circ$ denotes final states and $\bullet$ is a deadlock state. In this protocol, deadlock state is used to specify that the tourist has come to a decision and the common place would be the value of α when the deadlock state is reached.

Theorem 30. *In the concrete model, the specification of Rabin's protocol satisfies*

$$S = [((P + Q)\|M)^*((P + Q)\|C)^*]^*$$

Firstly, notice that if $\cdot$ is (conditionally) continuous then $x^* = \sup_{n\in\mathbb{N}}(1 + x)^n$. The proof relies on the fact that $f_x^n(0) = (1 + x)^n$ where $f_x(y) = 1 + x \cdot y$ and the result follows by taking the limit.

Proof. The above property allows us to express x^* as the limit of finite iterations of x interleaved with successful termination. We have

$$\begin{aligned}(P + Q)^*\|(M + C)^* &= \sup_m (1 + P + Q)^m \| \sup_n (1 + M + C)^n \\ &= \sup_m \sup_n [(1 + P + Q)^m \| (1 + M + C)^n]\end{aligned}$$

The processes P and Q are essentially delimited by $\alpha?K$ and $\alpha!K$ which ensures the following properties of the system

$$X \cdot A\|Y \cdot B = [X\|Y] \cdot [A\|B] \quad (17)$$
$$X \cdot A\|1 = 0 \quad (18)$$
$$Y \cdot B\|1 = 0 \quad (19)$$

for every processes A, B and where $X = P + Q$ is the collection of tourists and $Y = M + C$ is the collection of places.

In particular,

$$1\|(1+X)^n = 1\|(1+X)^{n-1} + 1\|X\cdot(1+X)^n = 1\|(1+X)^{n-1}$$

[12] A rigorous proof of this fact could be done by induction on the structure of x.

[13] A small difference from CSP is that we consider only words terminating to final states but since $F_{x\|y} = F_x \times F_y$, we are safe to use most of the general properties found in CSP such as

$$Tr(x\|y) = \{t \mid t_{|x} \in Tr(x) \wedge t_{|y} \in Tr(y)\}$$

.

and by induction, since $1\|1 = 1$,

$$1\|(1+X)^n = 1 \tag{20}$$

for every $n \in \mathbb{N}$. Similarly, $1\|(1+Y)^n = 1$.

On the other hand, let us denote $T_{m,n} = (1+X)^m\|(1+Y)^n$, then

$$\begin{aligned} T_{m,n} &= \left[(1+X)^{m-1} + X\cdot(1+X)^{m-1}\right] \| \\ &\qquad \left[(1+Y)^{n-1} + Y\cdot(1+Y)^{n-1}\right] \\ &= T_{m-1,n-1} + X\cdot(1+X)^{m-1}\|(1+Y)^{n-1} + \\ &\qquad (1+X)^{m-1}\|Y\cdot(1+Y)^{m-1} + \\ &\qquad [X\|Y]\cdot T_{m-1,n-1} \\ &= (1+X\|Y)\cdot T_{m-1,n-1} + U_{m-1,n-1} + \\ &\qquad V_{m-1,n-1} \end{aligned}$$

where

$$\begin{aligned} U_{m-1,n-1} &= X\cdot(1+X)^{m-1}\|[(1+Y)^{n-1} \\ &= U_{m-1,n-2} + [X\|Y]\cdot T_{m-1,n-2} \\ &= U_{m-1,n-3} + [X\|Y]\cdot T_{m-1,n-3} + \\ &\qquad [X\|Y]\cdot T_{m-1,n-3} \end{aligned}$$

Since the sequence $(1+Y)^n$ is monotone, $T_{m,n} \leq T_{m,n'}$ for every $n \leq n'$ and therefore $U_{m-1,n-1} \leq U_{m-1,0} + [X\|Y]\cdot T_{m-1,n-1}$. But Property 18 implies that $U_{m-1,0} = 0$.

Similarly, $V_{m-1,n-1} \leq [X\|Y]\cdot T_{m-1,n-1}$. Hence

$$T_{m,n} = (1+[X\|Y])\cdot T_{m-1,n-1}.$$

By induction, we show that

$$T_{m,n} = (1+[X\|Y])^{\inf(m,n)}$$

because $T_{0,n} = T_{m,0} = 1$ by Equation 20.

Finally, we have

$$\begin{aligned} X^*\|Y^* &= \sup_m \sup_n (1+X)^m\|(1+Y)^n \\ &= \sup_n (1+[X\|Y])^n \\ &= (X\|Y)^* \end{aligned}$$

□